Realizing Robustness Testing Based on TTCN-3

X.M. Liu
College of Information
Beijing City University
Beijing China

Y.P. Liu
College of Computer
Beihang University
Beihang China

S.M. Liu
College of Science and Information
Qingdao Agriculture University
Qingdao China

J. Wu
College of Computer
Beihang University
China

***Abstract*—Robustness refers to the ability to ensure the software system execute correctly under normal or abnormal conditions in software testing. Robustness testing needs a large number of illegal and effective inputs, which are one of the key researches. The data variance is based on mutation testing, which is a kind of white box testing through destructing the system under test for robustness testing. The variance data is a kind of determining the robustness by applying some interference to the test input to create abnormal data which is then used to the system under test. Because TTCN-3 is used for black box testing, it can get the data rule description from the system under test. The data variance is also used by TTCN-3 testing. Therefore, the automatic generation of robust illegal inputs is realized on the basis of the data variance in this paper.**

Keywords-robustness testing; data variance; TTCN-3; mutation testing

I. INTRODUCTION

The methods used in traditional software testing are mainly used to validate whether the software behaviours accord with demands, namely the function testing [1]. But the abnormal test is often not paid enough attention. With the software used in the important fields of politics, military and finance, the robustness of software has been paid more and more attention. Robustness de-scribes the degree of correctly running its functions of a system or a component in high intensity input environment or in an invalid data input [2]. As the input data from different system is different, the abnormal input of robustness is different too, which leads to the low reliability of robustness test case. At the same time, robustness testing needs the test case must cover all possible abnormalities or attack modes, the number of test cases is usually in large scale, which makes the robustness test case development more difficult [3].

To test the distributed systems and protocols is always an active research field by use of illegal input technology of robustness. Early studies focused on the illegal input technology of robustness based on hardware. Then the illegal input technology of robustness based on software (SWIFI) was introduced in order to reduce costs and the difficulty of development [4]. For SWIFI, the implementation technology and the method, describing faults of the illegal input of robustness, are the two main problems paid more attention during studying. A common method to implement the illegal input of robustness is to insert an additional illegal input layer of robustness which is responsible for injecting various communication faults between IUT and its underlying protocol. There are two options to insert the software [5]. EFA and Virtual wire select the specific location in the OS kernel to insert it, while Orchestra directly inserts it under IUT. Regardless of the way, the illegal input layer of robustness is usually developed as the kernel component OS. To describe the faults is by defining specific test scripts which can control test and the illegal input of robustness. The languages used by test scripts include TCL (such as Orchestra), C (such as EFA) and self-defined languages (such as FIAT and virtual wire). The disadvantage of test script is that its usability is not good which needs tester learn the syntax rules firstly [6]. At the same time, scripting manually also causes all kinds of errors easily. Based on the above analysis, the fault injection is commonly implemented by inserting the illegal input layer of robustness in the current research But it is closely related with the operation platform and OS, which leads to poor portability [7].

Therefore, a new technique to implement the illegal input of robust-ness is presented in this paper, which can effectively separate the independent parts from the system and improve its portability. Simultaneously, in order to solve the usability of test scripts, this paper presents the method to describe faults based on model. The fault activity is defined by the way of visualization in the test, which can automatically generate test scripts from the model and avoid writing the test scripts manually [8].

II. BUILDING THE ILLEGAL INPUT

Traditional software testing methods focus on verifying whether the software behaviours under test accords with demands, namely the function test. The abnormal condition test is not paid more attention. Different software has different input data, which makes the different illegal input of robustness and leads to the low reusability of test cases [9]. At the same time, the robustness testing needs that the test cases

must cover all possible abnormalities or attack modes, the number of test cases is usually in a larger scale, which makes the test cases development more difficult. The most typical method of robustness testing is to generate a large number of illegal inputs to test the SUT tolerance to abnormal data [10].

A. The Module Framework

In TTCN-3, the abstract layer is very high, whose test set is a kind of specification itself. So in this paper, mutation operators are defined based on TTCN-3 data types. The legal data unit defined in the set of TTC-3 test cases is made as input, which is mutated by selecting the mutation operator by way of some mutation strategy, the a large number of illegal input are produced. Therefore, the method to generate the illegal input of robustness testing is also based on specification. The module framework to generate the illegal input is shown as fig.1.

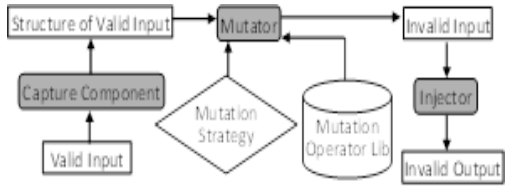

FIGURE I. THE MODULE FRAMEWORK TO GENERATE THE ILLEGAL INPUT

The capture component is used to obtain the grammatical structure of legal input. The Mutation component is used to generate illegal input by the mutation operator, according to the grammatical structure of legal data. The injector sends the illegal input to the SUT. In addition, the mutation operator library contains all the mutation operators that have already realized. The mutation strategy describes the rules to select the mutation operators. Both of them will serve as the input of Mutation component.

When generating the illegal input, it is very important to capture the legal input and analysis its se-mantic structure, which is related with selecting the mutation operator and implementing the mutation. There are two methods to capture the legal input. That is static analysis and dynamic capture. The static analysis is to analyse the abstract test set of TTCN-3 by using the parser and then extract semantics tree. The dynamic capture is to capture the calling of encode function by inserting codes between TE and CD. The parameter value of the function is the message instance with semantic structure, shown in fig.2.

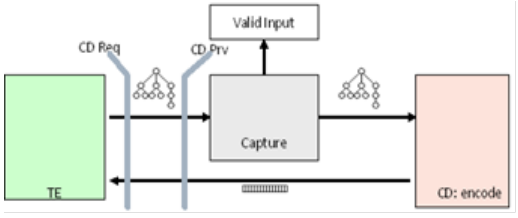

FIGURE II. CAPTURE THE LEGAL INPUT DYNAMICALLY

B. Select the Mutation Operator

The concept of mutation operator derives from mutation testing which is a kind of testing method oriented defects. The data variation method proposed in this paper is different from the mutation testing. Be-cause the data variation focuses on the data sent to the SUT, the mutation testing is a kind of program variation and focuses on the source code of the system under test. Their differences are shown in Table 1. Because the objects of the mutation operator are different, the factors concerned are different when designing the mutation operators.

TABLE I. THE DIFFERENCES BETWEEN THE MUTATION TESTING AND THE DATA VARIATION

Types of Mutation	the Objects of Mutation Operator	the Output of Mutation	Function
Mutation Testing (Program Variation)	the source code of the SUT	the variants of program	detect the adequacy of test
Data Variation	the input data of SUT	the abnormal input data	detect the robustness of SUT

The data variation focuses on motivating the SUT, so it is regarded as an operation with the sensitive data type. A message instance is made up of the structure and value which can be mutated, that is, breaking the data structure and updating the data value, or combining both of them. On the other hand, what types of mutation is applied to the data structure and the data value, it is also the concerned factors to design the mutation operation. The mutation operators designed in this paper and the TTCN-3 data type are shown in Table 2. To be explained, because the focus of this paper is the method to generate the illegal input based on the data variation, the design of mutation operator is not very comprehensive.

TABLE II. THE MUTATION OPERATOR CORRESPONDING TO THE DATA TYPE

Mutation Operator	Data Type	Remarks
Boundary Value	integer	be possible to lead to data overflow
++/--	integer	be possible to lead to data overflow
Insert the special characters repeatedly	char string	lead to the string buffer overflow
Insert the format characters repeatedly	char string	break the data string structure
Insert the special characters	char string	break the data string structure
Null string	char string	be possible to lead to the abnormal null pointe
Lack elements	record, set, record of, set of	be possible to lead to the abnormal null pointe
duplicate elements	record of, set of	lead o the abnormal data type
Elements disorder	record if	break the original order of data domain

Due to the different data types are corresponding the different mutation operators, and a mutation operation may adapt to a variety of data types, the mutation operation is determined according to the data type and then to be chosen before executing the data variation. So in this paper the

operation selection rule is described by use of variation strategy. The focus of this paper is not how to select the mutation operator, for ease of implementation, the mutation strategy used in this paper is the random selection. Firstly determine the set of mutation operator according to the data type and then select an operation from the set randomly.

Algorithm: Select the Mutation Operator.

INPUT: Operators: = {operatori| operatori is one of mutation operators in the mutation operator library, operator$_i$ ∈Operators}

InputType: = {field$_i$| field$_i$ is the element of valid input}

Map: = {<Operator, Type>}

OUTPUT: Operator: = {operator | operator ∈Operators}

ALGORITHM BEGIN

Set<Operator> operators;

Count: = 0;

FOR EACH entry<Operator, Type> IN Map

Set<Type> types = entry.getTypes (InputType);

IF InputType in types

operators.add (entry.getOperator ());

Count++;

END IF

END FOR

Index: = random(Count);

Operator: = operators.getOperator(Index);

RETURN Operator;

ALGORITHM END

In order to play the role of mutation strategy, the corresponding relationship model is built between the mutation operator and data type in this paper. The corresponding relation between them is based on the consistency of data type properties, which is described in XML, shown in fig.3.

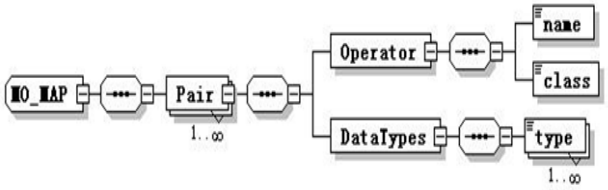

FIGURE III. THE SCHEMA OF THE RELATION BETWEEN THE MUTATION OPERATOR AND DATA TYPES

III. CASE STUDIES

The object under test is the TCP subsystem in this case. The subsystem receives messages sent by the client. Then the messages processed are then sent to the client. The data type of test data defined in the function test set of TCP subsystem is shown as follows:

type record tcpPacket {

charstring packetType length(2),

charstring packetLength length(2),

charstring packetData

}

During the test, the test data are mutated by use of the abnormal input generation module with robustness, the execution process is shown as fig.4.

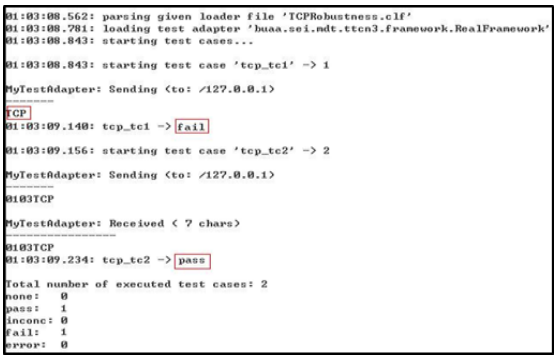

FIGURE IV. THE EXECUTION PROCESS OF ROBUSTNESS TESTING

In order to clearly observe the implementation process of data variation, the testing information col-lection module records the interaction information between the testing system and the SUT, shown in fig.5.

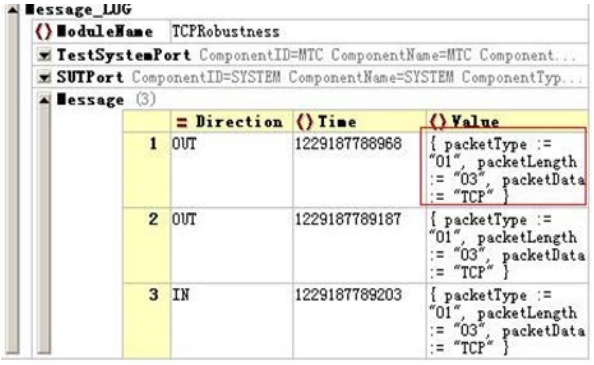

FIGURE V. THE INTERACTION INFORMATION LOG OF ROBUSTNESS TESTING

According to the test execution process and the message interaction records, the first test data is applied to the structured data types mutation operators, its packType and packLength lack values, which leads t the first data case execution results for fail. But then the test data is not mutated and the test is executed successfully, and obtains the correct response. It can be seen that the TCP subsystem under test has

a certain tolerance for the abnormal input with lacking data domain value.

IV. SUMMARY

The abnormal input generation module and the test information collection module with robustness greatly enhance the applicability of TTCN-3 testing technology. In this paper there are better innovations in implementation technology of the test adapter framework, the mapping relations between the test adapter entity and the SUT, and the method generating the abnormal input based on data variation.

The method generating the abnormal input based on TTCN-3 is one of the main researches in this pa-per, which can effectively improve the efficiency of developing TTCN-3 test system and reduce the test cost. During the implementation, it effectively uses many related technologies, such as reverse engineering, data modelling code generating, and it also achieved good results.

REFERENCES

[1] Dawson S, Jahanian F, Mitton T.ORCHESTRA: a fault injec-tion environment for distributed systems[C]//26th Int'l Symposium on Fault-Tolerant Computing (FTCS) , Jun 1996.

[2] Gábor Ziegler, György Réthy. Performance testing with TTCN-3 [R], TTCN-3 User Conference, 2006

[3] Du Wenliang, Mathur A P.Vulnerability Testing of Software System Using Fault Injection.Coast TR 98- 02, 1998.

[4] Gamma E, Kent Beck.Contributing to eclipse [M].[S.l.]: Adision Wesley, 2013.

[5] Arlat J, Aguera M, Crouzet Y, et al.Experimental evaluation of the fault tolerance of an atomic multicast system[J].IEEE Trans Reliability,1990, 39(4) : 455- 467.

[6] De P, Neogi A, Chiueh T- C.VirtualWire: a fault injection and analysis tool for network protocols[C]//Proc IEEE 23rd In-ternational Conference on Distributed Computing Sys-tems , Providence , Rhode Island, USA, 2013.

[7] Segall Z, Vrsalovic D, Siewiorek D P, et al.FIAT- Fault injection based automated testing environment [C]/ /Proc FTCS- 18 , Tokyo,Japan, 1988: 102- 107.

[8] FOKUS.Unified modeling language: testing profile, version 2.0.OMG Adopted Specification.Deutschland, 2004- 04.

[9] Rosenberg J, Schulzrine H, Camarillo G, et al.SIP: Session in-tiation protocol, RFC3261[S], 2012.

[10] Echtle K, Leu M.The EFA fault injector for fault- tolerant dis-tributed system testing[C]//Proc Workshop on Fault- Toler-ant Parallel and Distributed Systems, Amherst, USA, 2012.